

МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ  
ДЕПАРТАМЕНТ НАУЧНО-ТЕХНОЛОГИЧЕСКОЙ ПОЛИТИКИ И  
ОБРАЗОВАНИЯ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

# **Моделирование с помощью ArgoUML**



Красноярск  
2018

**Моделирование с помощью ArgoUML** [Текст]: Методические указания к выполнению лабораторных работ / сост.: Миндалёв И.В. -- Красноярск, Краснояр. гос. аграр. ун-т., 2018, 27 с.

Методические указания представляют собой руководство по проектированию UML-моделей с помощью CASE-средства ArgoUML.

Предназначено для студентов, изучающих дисциплины «Моделирование бизнес-процессов в агропромышленном комплексе», «Практика по получению профессиональных умений и опыта профессиональной деятельности», «Преддипломная практика» по направлению 09.03.03 «Прикладная информатика».

© Красноярский государственный аграрный университет, 2018

© Миндалёв И.В., 2018

## **1. Цель работы**

Создание современных ИС представляет собой сложнейшую задачу, решение которой требует применения специальных методик и инструментов. К ним относятся CASE-технологии и инструментальные CASE-средства позволяющие максимально систематизировать и автоматизировать все этапы разработки систем.

Целью выполнения практической работы является создание UML-модели приложения с требованиями языка UML с помощью CASE-средства ArgoUML.

## **2. Рекомендации по выбору предметной области**

Студент выбирает предметную область в соответствии с выданным номером варианта.

## **3. Требования к оформлению**

Пояснительная записка к практической работе должна быть оформлена в соответствии с действующими Государственными стандартами Российской Федерации и требованиями установленными в КрасГАУ [1].

Пояснительная записка является текстовым документом, который должен быть оформлен в соответствии с требованиями, установленными ГОСТ 2.105-79, ГОСТ 2.106-68.

Перенос слов на титульном листе не допускается. Точка в конце фраз не ставится. Структура титульного листа пояснительной записки приведена в приложении Г [1].

Пояснительная записка выполняется на листах белой бумаги формата А4 в соответствии с требованиями изложенными в [1].

## **4. Литература**

1 Матюшев В.В., Бастрон Т.Н., Шатурина Л.П.. Положение по оформлению текстовой и графической части учебных и научных работ (общие требования). polo\_kgau.pdf

19. Миндалев И.В. Моделирование бизнес-процесов. Электронно-методический комплекс. <http://enisey.name/umk/mbp/>

## **5. Программное обеспечение**

- CASE-средство ArgoUML.  
ArgoUML-0.32.2.zip
- Инструментальные средства разработки приложений Java Development Kit (JDK).  
jdk-6u20-ea-bin-b02-windows-i586-01\_apr\_2010.exe

## 6. Содержание работы

### 6.7. ArgoUML – средство средство UML моделирования. Краткий обзор возможностей.

ArgoUML — средство UML моделирования. ArgoUML является открытым программным обеспечением и распространяется под лицензией EPL.

ArgoUML полностью написан на Java и для работы ему подходит любая операционная система с установленной Java 2 JRE или JDK версии 1.4 или выше.

Функциональность ArgoUML включает в себя:

- Поддержку спецификаций UML 1.3, 1.4, XMI 1.0, 1.1, 1.2
- 9 видов диаграмм UML (диаграммы классов, состояний, кооперации, последовательности, деятельности, прецедентов, объектов, компонентов, развёртывания)
- Поддержку OCL для классов
- Генерацию исходного кода Java, C++, C# и PHP
- Обратный инжиниринг из исходного кода и байткода Java
- Автоматическую верификацию модели UML (design critics)

All 9 UML 1.4 diagrams supported though not yet implemented. Only class diagram and use-case diagrams are more or less fully implemented.

- Closely follows the UML standard.
- Platform independent – Java 1.5+.
- Click and Go! with Java Web Start (no setup required, starts from your web browser).
- Standard UML 1.4 Metamodel.
- [XMI](#) support.
- Export diagrams as [GIF](#), [PNG](#), [PS](#), [EPS](#), [PGML](#) and [SVG](#).
- Available in ten languages: EN, EN-GB, DE, ES, IT, RU, FR, NB, PT, ZH.
- Advanced diagram editing and zoom.
- Built-in design critics provide unobtrusive review of design and suggestions for improvements.

- Extensible modules interface.
- [OCL](#) support.
- Forward engineering (code generation supports C++ and C#, Java, PHP 4, PHP 5, Python, Ruby and, with less mature modules, Ada, Delphi and SQL).
- Reverse engineering / [JAR](#)/class file import.

## ArgoUML - Open Source Unified Modeling Language

<http://www.methodsandtools.com/tools/tools.php?argouml>

Franco Martinig, Martinig & Associates, [www.martinig.ch](http://www.martinig.ch)

ArgoUML is an open source UML modeling tool that includes support for all standard UML 1.4 diagrams. It runs on any Java platform and is available in ten languages.

**Web Site:** <http://argouml.tigris.org/>

**Version Tested:** ArgoUML 0.30.2, tested on Windows XP during November/December 2010

**System Requirements:** ArgoUML is a Java based application that needs Java 2 JRE or JDK version 1.4 or higher and 10MB of disk space

**License & Pricing:** Open Source, Eclipse Public License (EPL) 1.0.

**Support:** User forum: <http://www.argouml-users.net/forum/>

### Installation

ArgoUML can be installed using the Java Web Start procedure connected to ArgoUML home page. For Windows you can also download a setup file that will install it in 30 seconds and launch the application, creating a desktop icon.

### Documentation

The documentation (<http://argouml.tigris.org/documentation/index.html>) is impressive with different formats of a quick start and a user manual (403 pages!) that are available in English, Spanish and German. Additional ArgoUML extensions documentation and UML resources are also listed in the documentation section. An on-line tool tour allows getting a quick overview of ArgoUML interface and features.

### Configuration

The configuration of ArgoUML is separated in different places. You have a classical "settings" menu where you are able to configure the user interface options like language or appearances. It is also there that you enable or disable the associated modules, like the Java code generator and some of their settings. The "Critique" menu allows to toggle design critics that create "todos" elements during your modeling activity and to adjust the importance of critics.

## Features

ArgoUML support the following UML 1.4 diagram types:

- Class diagram
- Statechart diagram
- Activity diagram (including Swimlanes)
- Use Case diagram
- Collaboration diagram
- Deployment diagram (includes Object and Component diagram in one)
- Sequence diagram

ArgoUML also provides code generation for Java, C++, C#, PHP4 and PHP5. It also enables reverse engineering from Java. External modules have been developed to complement ArgoUML in specific areas. They provide generation of database schemas or code in other languages like Ruby or Delphi. You can find a list of most of these modules on

<http://design.tigris.org/>

In this evaluation, we will concentrate only on the UML diagramming features of ArgoUML.

## User Interface

The screen of ArgoUML is split in four different panes. The "explorer" pane shows relationships between diagrams and design items according to the selected perspective. The "ToDo" pane contains the tasks that could be completed. The main window is the drawing window where you create your diagrams. On the bottom, you find a "details" pane where you can define your diagram items and link them with elements, like a "todo" item or documentation.

## Drawing Diagrams

The modeling process is rather intuitive and smooth, The only missing feature is an "undo/redo" capability. The mouse gives you hints on each

element that you can place on your diagram. You can then use the detail pane to describe diagram items and link to other items like documentation for instance. After each action, your model is assessed and the "todo" panel on the bottom left is updated. Clicking on one of this item will give you an explanation about its rationale and how you should act to improve your design. You can naturally turn off this constant evaluation of your design.

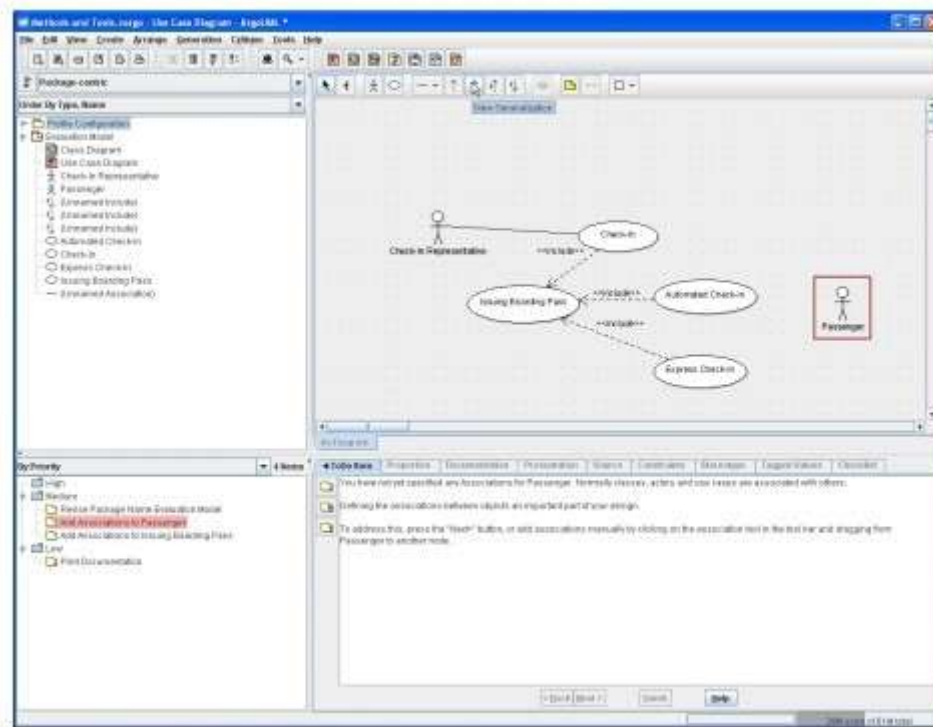


Figure 1. Design assessment during the construction of a Use Case Diagram

Another interesting feature of ArgoUML is the presence of checklist for every component of a model.

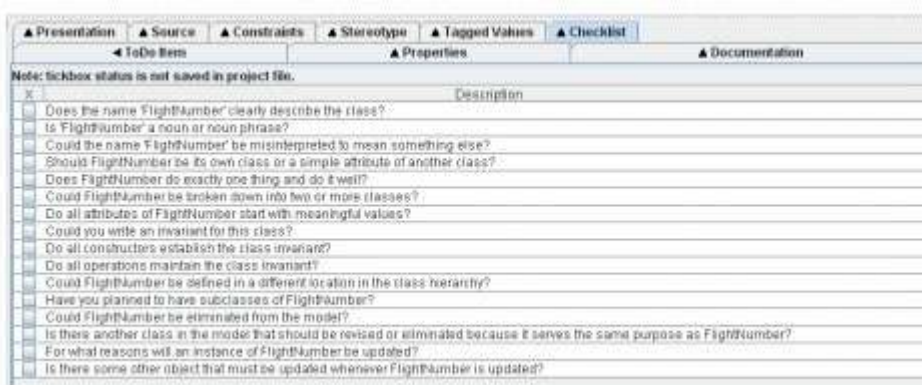


Figure 2. Checklist for a "FlightNumber" class  
Class Models



Class models are influenced by generation. You can see the generated results as you are building your class and the generating process influences the design assistance. Diagrams can be graphically exported in the various formats, so that you can include them in other documents.

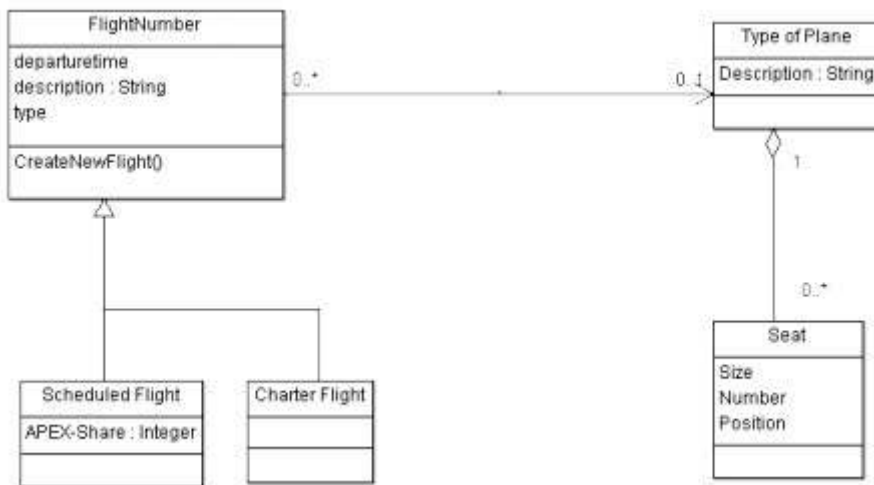


Figure 3. Exported Class Diagram

## Conclusion

ArgoUML is an active open source project that provides a working tool to support basic UML modeling activities. The diagramming features are easy to use and provides useful assistance in the formatting / aligning process. The nicer aspects of ArgoUML lie in its design assistance features. The design evaluation and checklists provides valuable help to make sure that your models are well formed. This will be specifically attractive for people that are learning UML diagrams or don't use them continuously. Beside the modeling aspects, ArgoUML has also some nice features like code generation and reverse engineering. I haven't checked these aspects, but if you develop with Java you might be interested to explore them.

## Reference

Models of this evaluation inspired by the book "UML 2.0 in Action", Patrick Grдssle, Henriette Baumann, Philippe Baumann, Packt Publishing, ISBN 1-904811-55-8

## 6.8. Где скачать?

Для свободного скачивания доступен ArgoUML.

<http://argouml.tigris.org/>

- The home of UML: [The Object Management Group \(OMG\)](#)
- UML specifications: [UML 2.2](#), [UML 1.4](#)

## 6.9. Установка JDK



Зачем? Для работы многих программ, в том числе Data Modeler, Oracle SQL Developer, Ramus Education необходимы JRE или JDK.

Java, JRE, JDK? Что это такое? Java — объектный язык программирования, разработанный компанией Sun на основе языка C++. Компания предоставляет на своем сайте для свободного доступа спецификацию языка (описывающая лексику, типы данных, основные конструкции) и инструментальные средства разработки приложений — Java Development Kit (JDK).

Главная задача, которую преследовали разработчики — создание надежного платформо-независимого объектного языка, который позволял бы разрабатывать небольшие мобильные приложения для web — апплеты. Технология разработки и использования java-апплетов основана на двух стандартизованных компанией компонентах: на мобильных Java-байт кодах — формате представления результатов компиляции исходного программного кода java-апплета — и на спецификации виртуальной машины Java (JVM) — интерпретаторе мобильных java-байт кодов. Скомпилированные апплеты должны храниться на web-сервере. Их вызов на машину клиента обеспечивается браузером при просмотре HTML-страницы, в которой с помощью специальных тегов встроен вызов апплета с передачей ему фактических параметров. После вызова мобильных java-байт кодов на сторону web-клиента их исполнение осуществляется JVM, встроенной в браузер.

Наряду с использованием java для создания апплетов широко используются системы программирования с входным языком java, основанные на традиционной технологии.

Интерес к Java был также обусловлен появлением таких технологий, как J2EE, включая JSP (Java Server Pages), J2ME сделавших Java наиболее популярной платформой для создания корпоративных решений, поддерживаемой почти всеми производителями ПО. Основная

сфера применения Java — это приложения масштаба предприятия и многозвенные распределенные системы, базирующиеся на J2EE-совместимых серверах приложений.

15 июня 1999 г. Sun объявила о разделении развития платформы Java 2 на три направления: Java 2 Platform Standard Edition (J2SE); Java 2 Platform Enterprise Edition (J2EE); Java 2 Platform Micro Edition (J2ME).

J2SE предназначается для использования на рабочих станциях и ПК. Standard Edition — основа технологии Java и прямое развитие JDK (средство разработчика было переименовано в j2sdk).

J2EE содержит все необходимое для создания сложных, высоконадежных, распределенных серверных приложений. Enterprise Edition — это набор мощных библиотек (например, Enterprise Java Beans, EJB) и пример реализации платформы (сервера приложений, Application Server), которая их поддерживает.

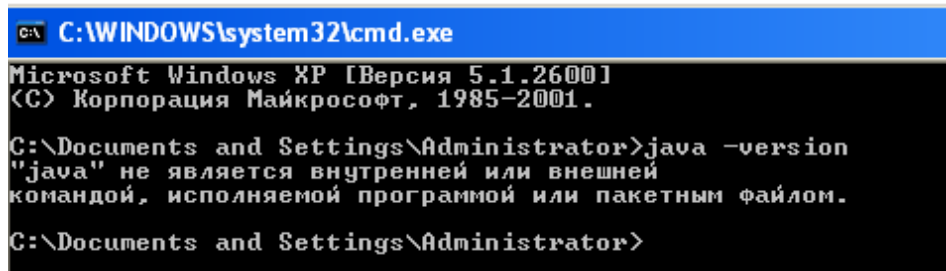
J2ME является усечением Standard Edition, чтобы удовлетворять жестким аппаратным требованиям небольших устройств, таких как карманные компьютеры и сотовые телефоны. По мнению аналитической компании Gartner Group, к 2004 году с вероятностью 0,7 на J2ME будут основаны 37% приложений для PDA (Personal Digital Assistant), а к 2005-му с той же вероятностью 65% сотовых телефонов будут оснащены виртуальной Java-машиной.

11 марта 1997 года Sun начала предлагать Java Runtime Environment (JRE) — среду выполнения Java. Это минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки. Если пользователь хочет только запускать программы, это именно то, что ему нужно.

JDK долгое время было базовым средством разработки приложений. Оно не содержит никаких текстовых редакторов, а оперирует только уже существующими java-файлами. Компилятор представлен утилитой javac (java compiler). Виртуальная машина реализована программой java. Для тестовых запусков апплетов существует специальная утилита appletviewer. Наконец, для автоматической генерации документации на основе исходного кода прилагается средство javadoc.

**Где взять?** Комплект разработчика **JDK** бесплатно распространяется с сайта <http://java.sun.com/javase/downloads/index.jsp>

6. Запустите Java и узнайте ее версию: выберите **Пуск, Программы, Выполнить**, введите команду **cmd**, в открывшейся консоли введите команду **java -version**.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\Documents and Settings\Administrator>java -version
"java" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Documents and Settings\Administrator>
```

Рис. 1:

Если сообщение как на рис. 1, то необходимо установить java, то есть выполняем пункт 2. Если сообщение примерно как на рис. 6, то java уже установлена.

7. Запустите установку: выберите **jdk-6u20-ea-bin-b02-windows-i586-01\_apr\_2010.exe, M2**.

8. Разработчики программы приветствуют вас, выберите **Next**.

9. В окне **License Agreement** (рис. 2) предлагается ознакомиться с условиями лицензионного соглашения, выберите **Accept >**.

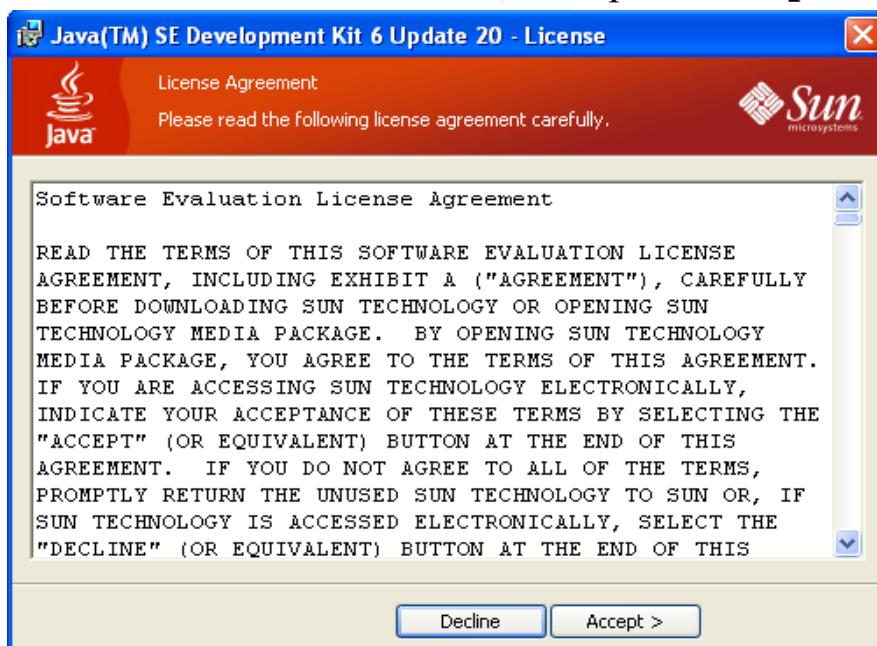
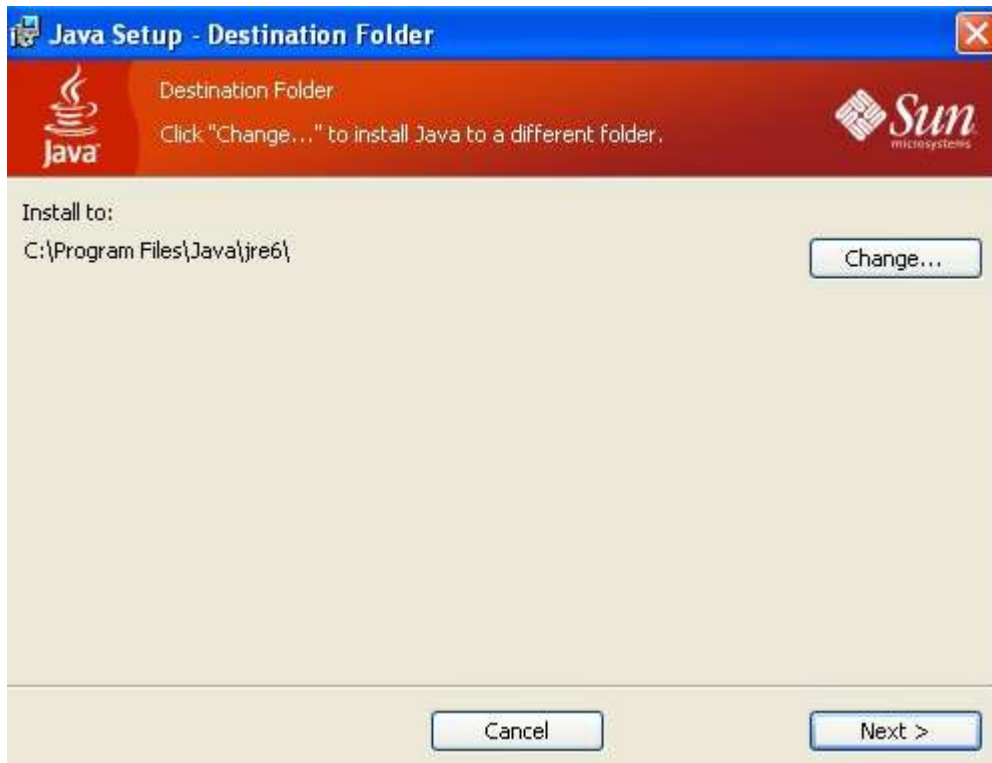
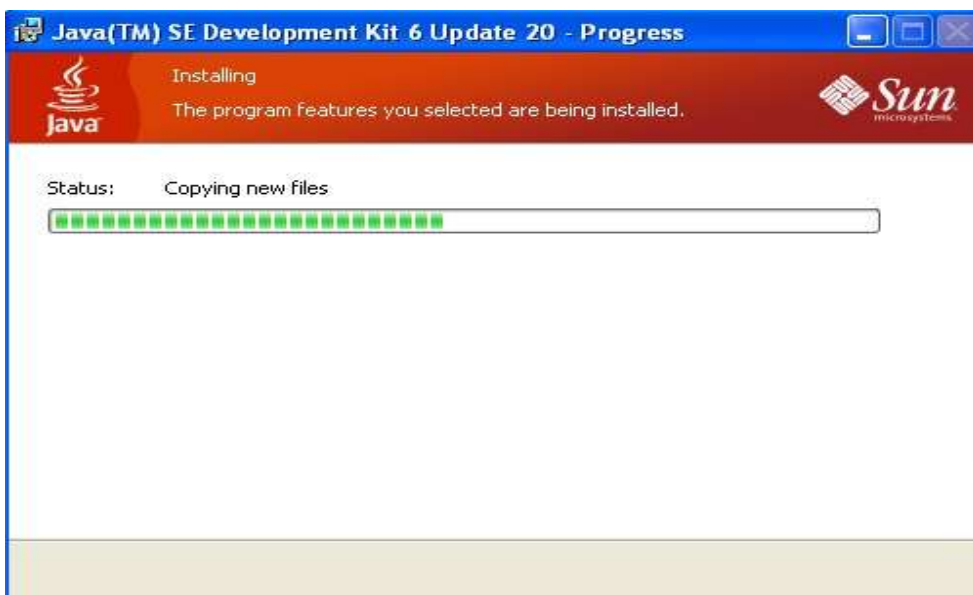


Рис. 2:

10. В окне **Custom Setup** (рис. 3) предлагается выбрать программы для установки, выберите по умолчанию, то есть все, обратите внимание на каталог **C:\Program Files\Java\jdk1.6.0\_20** где будет размещаться приложение, нажмите **Next**. Начинается установка.



Puc. 3:



Puc. 4:

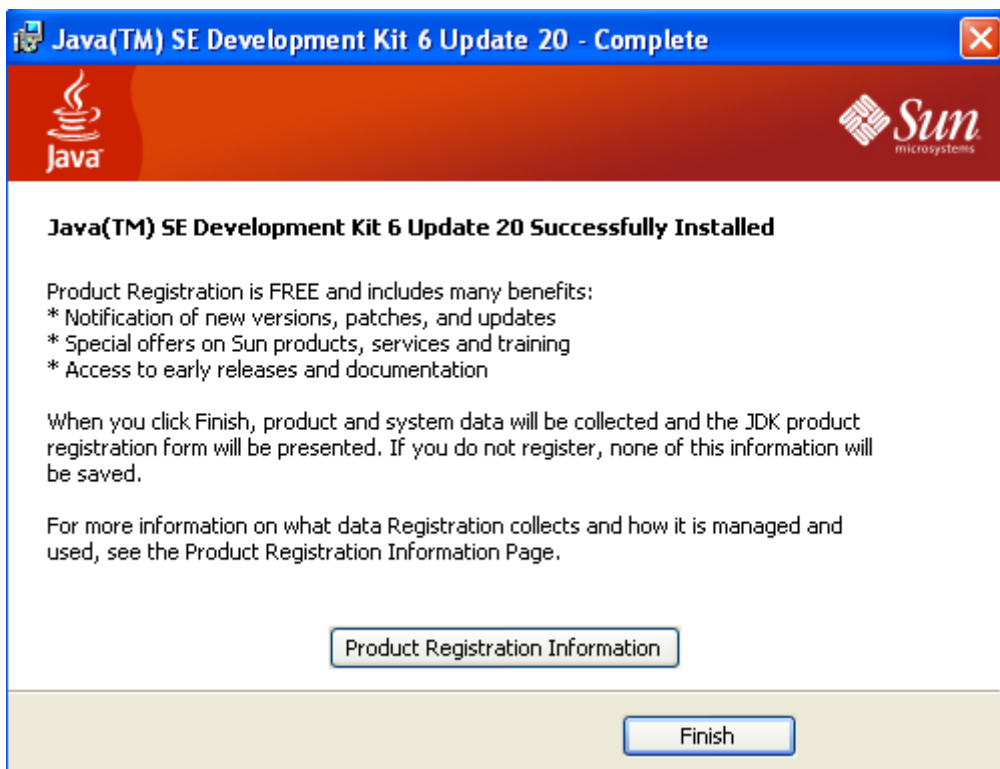


Рис. 5:

11. . После завершения установки: выберите **Finish**.

12. Войдите в каталог **jdk1.6.0\_20**, там должны быть следующие папки: **bin** (здесь находятся основные утилиты, например компилятор и JVM); **demo** (примеры программ); **jre** (файлы, имеющие отношение непосредственно к JRE); **lib** (библиотеки JDK). Кроме того здесь также должен быть файл **src.zip**. В нем находятся исходные кода библиотек, которые вы при желании можете изучить самостоятельно.

13. Перезагрузите компьютер.

14. Проверьте версию Java: выберите **Пуск | Программы | Выполнить**, наберите **cmd**, в открывшейся консоли введите команду: **java -version**

Если результат примерно такой, то можно себя поздравить — все установлено правильно.

```
Command Prompt
Microsoft Windows XP [Версия 5.1.2600]
(C) Корпорация Майкрософт, 1985-2001.

C:\Documents and Settings\Administrator>java -version
java version "1.6.0_20-ea"
Java(TM) SE Runtime Environment (build 1.6.0_20-ea-b02)
Java HotSpot(TM) Client VM (build 17.0-b12, mixed mode, sharing)

C:\Documents and Settings\Administrator>_
```

Рис. 6:

**6.10. Установка ArgoUML**

**6.11. Построение UML-моделей**



## Элементы интерфейса\*

1. Запустите CASE-средство: откройте **C:\ArgoUML-0.19.6\**, выберите **argouml.jar**, нажмите M2.

2. Найдите пять основных элементов интерфейса ArgoUML:

- **Меню и панели инструментов** – применяются для быстрого доступа к наиболее распространенным командам;
- **Навигатор** – используется для быстрой навигации по модели;
- **Окно редактирования** – используется для просмотра и редактирования диаграммы UML;
- **Окно деталей** – позволяет редактировать различные детали модели;
- **Окно To-Do** — The To-Do Pane displays the items on the models to-do list in a tree which sorts the list in a number of different ways.

## Варианты использования и действующие лица

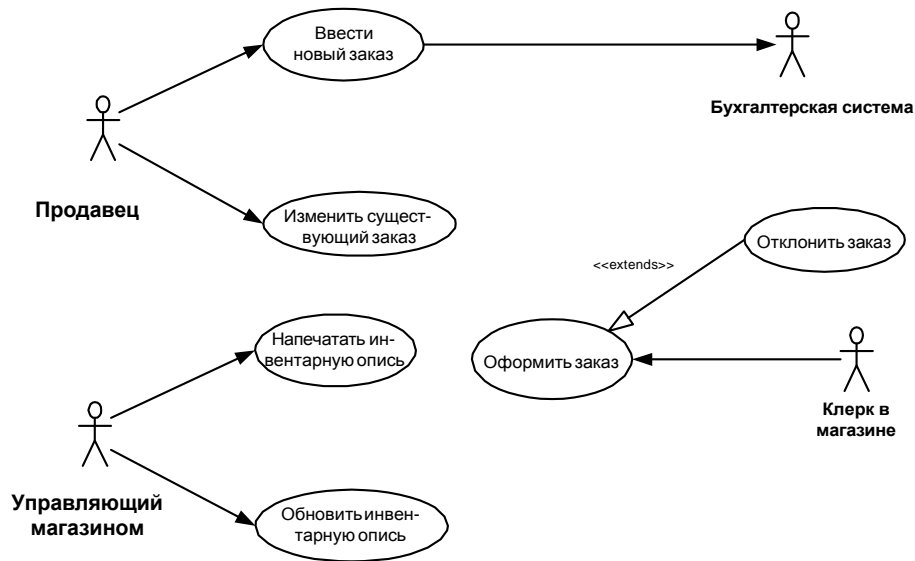


Рис. 1. Диаграмма вариантов использования для системы обработки заказов

## Постановка задачи

— Опять! — сказал Коля, повесив телефонную трубку. Оля взглянула на него, оторвавшись от компьютера:

\* © И.В. Миндалев, 2006

© Красноярский государственный аграрный университет, 2006

Данная публикация охраняется законом РФ «Об авторских и смежных правах» (ред. Федерального закона 19.07.95 № 110-ФЗ).

При использовании и цитировании ссылка на «Образовательный центр кафедры ИС и технологий в экономике КГАУ» обязательна.

При использовании и цитировании в интернете гиперссылка на <http://www.kgau.ru/istiki> обязательна.

— В чем дело?

— Четвертый раз за этот месяц один из наших клиентов жалуется, что не получил своего заказа. Если так будет продолжаться и дальше, мы вылетим из дела.

— Остынь, — ответила Оля. — Мы просто слишком быстро растем. Обработка всех заказов на бумаге проходила прекрасно, когда наша компания состояла из пяти человек. Нельзя ожидать от нее такой работы теперь. Давай поговорим со Натали, может быть, мы сумеем спроектировать систему, которая поможет нам управиться со всем этим.

“Шкафф” — это маленькая компания, специализирующаяся на производстве кухонных шкафов. Компания сформировалась три года назад из небольшой группы собравшихся вместе предпринимателей. Тогда поступало слишком мало заказов, и с ними вполне можно было управляться на бумаге. С ростом репутации компании число заказов возрастало. Пришлось нанять новых рабочих, и за три года фирма выросла до магазина с более чем 50 сотрудниками.

Теперь уже нельзя полагаться на обработку заказов вручную. Владельцы фирмы Коля и Оля решили поговорить со Натали, чтобы решить эту проблему. Натали — специалист по компьютерам. Она работает в отделе этой фирмы, занимающемся информационными технологиями.

Коля пошел звонить Натали:

— Совершенно очевидно, что нам требуется система по обработке заказов. Мы столкнулись с серьезным риском потерять клиентов.

— Согласна.

— Можешь ли ты разработать программу на Java, которая отслеживала бы заказы?

— Пока не волнуйтесь по поводу реализации. Давайте решим, чего вы хотите от системы.

— Она должна отслеживать заказы.

— Не мог бы ты быть более конкретным? Давай рассмотрим нынешний процесс.

— Хорошо. Получив звонок, мы заполняем форму заказа и передаем ее Олеся в магазин. Олеся заполняет все необходимые документы и готовит отправку товара клиенту. Копию формы мы отдаем Марине в бухгалтерию. Она вводит ее в бухгалтерскую систему и выписывает счет.

— И вы хотите, чтобы новая система поддерживала весь этот процесс?

— Точно.

Из этого разговора Натали смогла понять, что система должна обеспечивать возможность добавления новых заказов, изменения старых, выполнения заказов, проверки и возобновления инвентарных описей. При получении заказа система должна послать сообщение бухгалтерской системе, которая выписывает счет. Если требуемого товара нет на складе, заказ должен быть отклонен. Затем Натали преобразовала требования в диаграмму Вариантов Использования, с помощью которой начала моделировать систему.

3. Сохраните модель: выберите **Файл | Сохранить как**, в списке **Files of Type** выберите **\*.zargo**, в поле **File Name** введите **шкафф\_фузеева** (где вместо фузеева ваша фамилия), нажмите **Save**.

## Работа с критиками проекта

В то время как вы работаете над своей моделью, ваша работа проверяется непрерывно и незримо программой называемой критиком проекта. Он подобен персональному наставнику, который наблюдает за вами и уведомляет каждый раз когда видит кое-что сомнительное в вашем проекте.

4. Посмотрите что советует критик: в **окне To-Do** выберите папку **Средний**, M2, выберите сообщение **Revise...** и читайте в окне **Детали**, что обычно, имена пакета пишутся в строчных буквах. А заданный по умолчанию пакет верхних уровней, созданный ArgoUML называется **untitledModel** и поэтому нарушает принцип проекта.

5. Выберите **Следующий**, в поле **Name** введите **обработказаказов**, нажмите **Закончить**, критика исчезла.

6. Выберите вкладку **Документация**, в поле **Автор** введите свою фамилию.

7. В **окне To-Do** выберите сообщение **Add Elements...** и читайте в окне **Детали**, что необходимо создавать элементы модели и размещать их на диаграмме. В навигаторе выберите **Диаграмма вариантов использования 1**, нажмите **Новый вариант использования**, мышью выберите на окне **редактирования**, сообщение **Add Elements...** исчезло, но появилось другое.



## Создание вариантов использования

8. Создайте вариант использования: выберите мышью в браузере **Диаграмма вариантов использования 1**, выберите мышью на панели инструментов **Новый вариант использования**, выберите мышью в окне редактирования место размещения варианта использования.

9. В навигаторе должен появиться новый вариант использования под названием **anon UseCase**. Слева от его имени вы увидите пиктограмму варианта использования UML. Обратите внимание, что в окне To-Do появилось критика: выберите **Choose a Name**, в окне **Детали** выберите **Следующий**, в поле **Имя** введите его имя **Вести новый заказ**, нажмите **Закончить**.

10. Исходя из потребностей действующих лиц создайте следующие варианты использования: **Изменить существующий заказ**, **Напечатать инвентарную опись**, **Обновить инвентарную опись**, **Оформить заказ**, **Отклонить заказ**.

## Создание действующих лиц

11. Создайте действующее лицо: выберите мышью в браузере **Диаграмма вариантов использования 1**, выберите мышью на панели инструментов **Новый актер**, выберите мышью в окне редактирования место размещения актера.

12. В навигаторе должно появиться новое действующее лицо под названием **anon actor**. Слева от его имени вы увидите пиктограмму действующего лица UML. Обратите внимание, что в окне To-Do появилось критика: выберите **Choose a Name**, в окне **Детали** выберите **Следующий**, в поле **Имя** введите его имя **Продавец**.

13. Создайте следующие действующие лица: **Управляющий магазином**, **Клерк магазина**, **Бухгалтерская система**.

## Создание абстрактного варианта использования

14. Щелкните мышью на варианте использования **Отклонить заказ** на диаграмме, в окне **Детали** в поле **Модификаторы** выберите **Абстрак**, чтобы сделать этот вариант использования абстрактным.

### *Добавление ассоциации*

Обратите внимание, что в окне To-Do появилось куча критика: **Add Association**

15. С помощью кнопки **Новая ассоциация** панели инструментов создайте ассоциацию между действующим лицом **Продавец** и вариантом использования **Вести новый заказ**.

16. Создайте направление связи: выберите созданную **ассоциацию**, нажмите **МП**, выберите **Навигируемость | Продавец to вести новый заказ**.

17. Создайте остальные ассоциации — см. Рис. 1

Обратите внимание, что в окне To-Do критика исчезает.

### *Добавление связи расширения*

18. С помощью кнопки **Новое обобщение** (Generalization) панели инструментов создайте связь между вариантом использования **Отклонить заказ** и вариантом использования **Оформить заказ**.

Стрелка должна идти от первого варианта использования ко второму. Связь расширения означает, что вариант использования **Отклонить заказ** при необходимости дополняет функциональные возможности варианта использования **Оформить заказ**.

19. Выберите мышью на новой связи между вариантами использования **Отклонить заказ** и **Оформить заказ**, в окне **Детали** на вкладке **Свойства** нажмите кнопку **Новый стереотип** в поле **Имя** введите слово **extends** (расширение), затем щелкните мышью на окне редактирования, в поле **Имя** введите **1** надпись << **extends** >> появится на линии данной связи.

### *Добавление описаний к вариантам использования*

20. Выделите в навигаторе вариант использования **Вести новый заказ**, в окне **Детали** выберите вкладку **Документация**, в поле **Документация** введите следующее описание: **Этот вариант использования дает клиенту возможность ввести новый заказ в систему**.

21. Добавьте описания ко всем остальным вариантам использования.

### *Добавление описаний к действующему лицу*

22. Выделите в навигаторе действующее лицо **Продавец**, в окне **Детали** выберите вкладку **Документация**, в поле **Документация** введите следующее описание: **Продавец — это служащий, старающийся продать товар**.

23. Добавьте описания к остальным действующим лицам.

## Взаимодействие объектов

### Постановка задачи

Поговорив с Колей, Натали поняла, что должна делать система обработки заказов, создаваемая ею для фирмы Шкафф. Она нарисовала диаграмму Вариантов Использования. Изучив эту диаграмму, все пришли к согласию по поводу области применения системы.

Теперь наступило время анализа ее составных частей. Высший приоритет среди пользователей имеет вариант использования "Вести новый заказ", он же связан с наибольшим риском. Натали решила заняться им в первую очередь.

Она поговорила с Максимом, заведующим отделом продаж. Вдвоем они обсудили поток событий, который будет реализовываться в варианте использования. Получив нужную информацию, Натали составила описание сценариев:

Продавец вводит новый заказ. Продавец пытается ввести заказ, но товара нет на складе. Продавец пытается ввести заказ, но при его сохранении в базе данных возникает ошибка. Затем она приступила к созданию диаграмм Последовательности и Кооперативных диаграмм для сценария "Вести новый заказ".

### Создание диаграммы последовательности

Создайте диаграмму Последовательности, отражающую ввод нового заказа в систему обработки заказов. Это только одна из диаграмм, необходимых для моделирования варианта использования "Вести новый заказ". Она соответствует успешному варианту хода событий. Для описания того, что случится, если возникнет ошибка или если пользователь выберет другие действия из предложенных, придется разработать дополнительные диаграммы. Каждый "альтернативный поток варианта использования может быть промоделирован с помощью собственных диаграмм Взаимодействия.

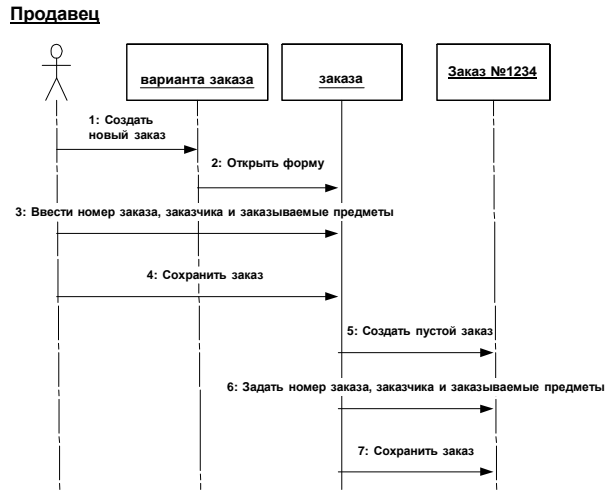


Рис. 2. Диаграмма Последовательности ввода нового заказа после завершения первого этапа работы

24. Создайте диаграмму последовательности: выберите мышью в навигаторе вариант использования **Ввести новый заказ**, выберите Создать диаграмму | Диаграмма последовательности (Sequence Diagram), в окне Детали в поле **Имя** введите **Диаграмма последовательности ввод заказа**.

### Добавление на диаграмму действующего лица и объектов

25. Добавьте на диаграмму действующее лицо: выберите диаграмму **Ввод заказа**, перетащите действующее лицо **Продавец** из браузера на созданную диаграмму. Не получилось? Диаграмма последовательности введена разработчиками AgroUML только в версии 0.19 и похоже не все еще работает.

26. Тогда сделайте не корректную операцию: нажмите кнопку **New Classifier Role** панели инструментов, щелкните мышью в верхней части диаграммы, чтобы поместить туда новый объект, в поле **Имя** введите **Продавец**.

27. Поместите на диаграмму объекты: **Форма выбора варианта заказа**, **Форма детали заказа**, **Заказ №1234**

### Добавление сообщений на диаграмму

28. На панели инструментов нажмите кнопку **Действие вызова**, проведите мышью от линии жизни **Продавец** к линии жизни объекта **Форма выбора варианта заказа**, введите имя **Создать новый заказ**.

29. Поместите на диаграмму остальные сообщения: см. **Рис. 2**

Завершен первый этап работы. Готовая диаграмма Последовательности представлена на рис. 2. Теперь нужно позаботиться об управляющих объектах и о взаимодействии с базой данных. Как видно из диаграммы, объект **Детали заказа** имеет множество ответственностей, с которыми лучше всего мог бы справиться управляющий объект. Кроме того, новый заказ должен сохранять себя в базе данных сам. Вероятно, эту обязанность лучше было бы переложить на другой объект.

### Добавление на диаграмму дополнительных объектов

30. Используя кнопку **New Classifier Role** панели инструментов введите новые объекты —

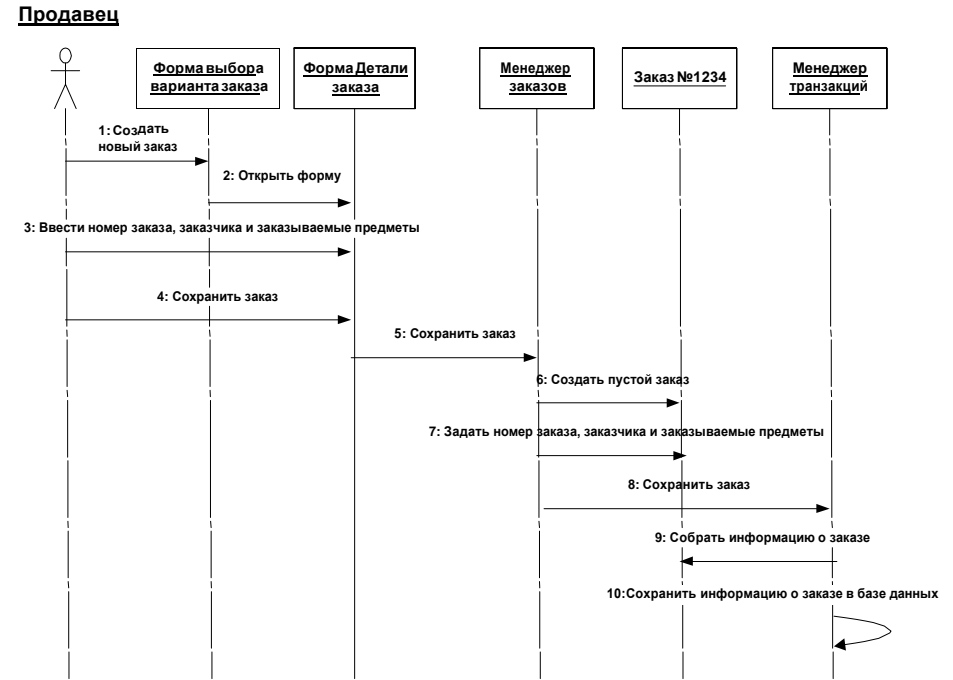


Рис. 3. Диаграмма Последовательности с новыми объектами

Менеджер заказов и Менеджер транзакций — см. рис. 3

31. Отредактируйте диаграмму по **Рис. 3**.

## Создание Кооперативной диаграммы

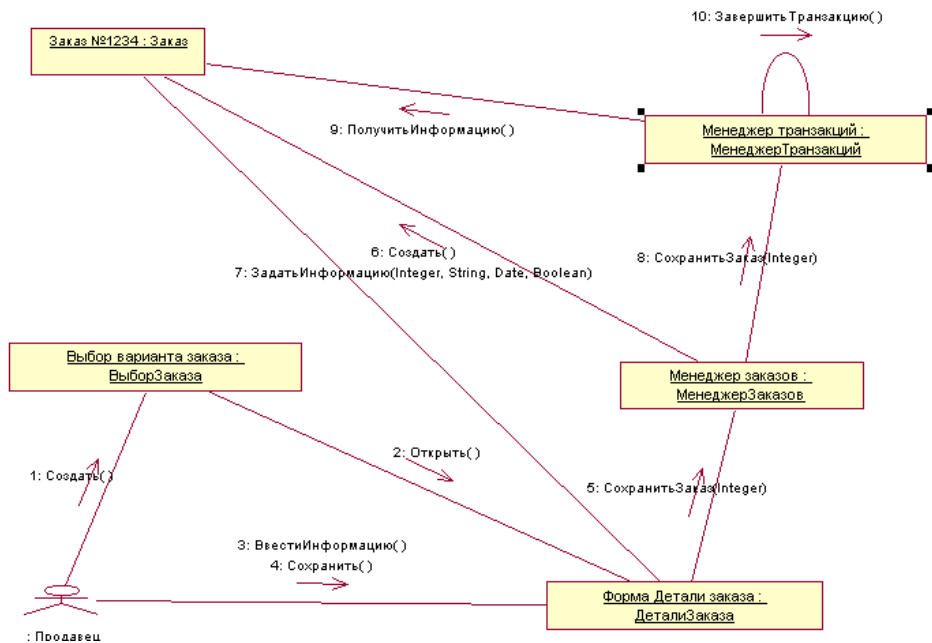
32. Создайте кооперативную диаграмму: выберите в навигаторе конфигурацию по диаграмма, выберите мышью в навигаторе вариант использования **Ввести новый заказ**, выберите Создать диаграмму | Диаграмма коопераций (Collaboration Diagram), в окне Детали в поле **Имя** введите **Диаграмма коопераций ввод заказа**.

33. В навигаторе найдите объекты **Форма выбора варианта заказа**, **Форма детали заказа**, **Заказ №1234**, **Продавец**, **Менеджер заказа**, **Менеджер транзакций** и перетащите их на диаграмму кооперации.

34. Определите направление связи — см. **Рис. 4**.

35. Имена связей соотнесите с операциями — см. **Рис. 4**

Рис. 9. Кооперативная диаграмма с показанными на ней операциями



### 3.1. Постановка задачи

Изучив диаграммы Взаимодействия, Коля понял, что система соответствует требованиям компании. После этого Натали пришла к руководителю группы разработчиков Ане:

- Вот диаграммы Взаимодействия, описывающие процесс ввода нового заказа.
- Прекрасно. Приступаем к разработке.

Ознакомившись с классами модели Rose, Аня решила объединить их в пакеты по стереотипу. Она создала пакеты Entities (Сущности), Boundaries (Границы) и Control (Управление), поместив в них соответствующие классы. Затем для каждого пакета были построены диаграммы Классов. Кроме того, на Главной диаграмме были показаны пакеты, а на диаграмме "Ввод нового заказа" — все классы этого варианта использования.

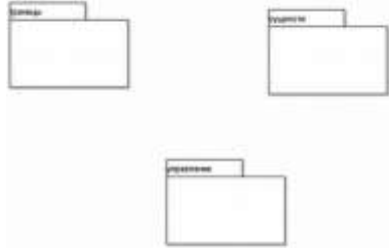
### 3.10. Постановка задачи

После добавления к классам атрибутов и операций Аня была уже почти готова к генерации кода. Сначала, однако, она должна была изучить связи между классами. Чтобы найти связи, Аня изучила диаграммы Последовательности. Все взаимодействующие там классы нуждались в определении соответствующих связей на диаграммах Классов. После обнаружения связей Карен добавила их в модель.

## Упражнение 3. Создание диаграмм классов

### 3.2. Создание пакетов

36. Запустите CASE-средство: откройте `C:\ArgoUML-0.19.6\`, выберите `argouml.jar`, нажмите **M2**.
37. Откройте модель: выберите **Файл | Открыть проект**, выберите **шкаф2**
38. В навигаторе сконфигурируйте перспективы **По диаграммам**.
39. Выберите диаграмму **вариантов использования 1**, просмотрите ее.
40. Выберите кооперативную диаграмму **Ввод заказа (кооп)**, просмотрите ее.
41. Выберите диаграмму последовательности **Ввод заказа (послед)**, просмотрите ее.
42. В навигаторе выберите сортировку **По пакетам**.
43. Создайте пакет: в навигаторе выберите модель **обработказаказов**, нажмите **МП**, в открывшемся меню выберите пункт **add Package**, в окне **Детали** в поле **Имя** введите **сущности**.
44. Создайте пакеты **границы** и **управление**.



### 3.3. Создание Главной диаграммы классов

45. Выберите мышью в навигаторе диаграмму классов **1**, в поле **Имя** введите **Главная (класс)**.
46. Перетащите пакет **сущности** из навигатора на диаграмму **Главная (класс)**.
47. Перетащите пакеты **границы** и **управление** из навигатора на ту же диаграмму. Результат на **Рис. 1**

### 3.4. Создание диаграммы классов

48. Создайте диаграмму классов: выберите в навигаторе вариант использования **Ввести новый заказ**, выберите **Создать диаграмму | Диаграмма классов**, в окне **Детали** в поле **Имя** введите **Ввод заказа (класс)**.

49. На диаграмме **Ввод заказа (класс)** с помощью кнопки **Новый класс** создайте классы **ВыборЗаказа**, **ДеталиЗаказа**, **МенеджерЗаказов**, **МенеджерТранзакций**, **Заказ**.

### 3.5. Добавление стереотипов к классам

50. Выберите класс **ВыборЗаказа**, в окне **Детали** нажмите кнопку **Вверх**, выберите кнопку **Новый стереотип**, в поле **Имя** введите **граница**, в списке **Базовый класс** выберите **Class**.

51. Выберите класс **ДеталиЗаказа**, в окне **Детали** в поле **Стереотип** выберите **граница**. 52. Свяжите классы **МенеджерЗаказов**, **МенеджерТранзакций** со стереотипом **управление**, а класс **Заказ** — со стереотипом **сущность**. Результат на **Рис. 2**.

### 3.6. Объединение классов в пакеты

53. В навигаторе перетащите класс **ВыборЗаказа** на пакет **границы**, перетащите класс **ДеталиЗаказа** на пакет **границы**, перетащите классы **МенеджерЗаказов**, **МенеджерТранзакций** на пакет **управление**, перетащите класс **Заказ** на пакет **сущности**.

### 3.7. Добавление диаграмм классов к каждому пакету

54. В навигаторе выберите диаграмму **Главная (класс)**, выберите пакет **границы**, **M2**, на запрос о создании диаграммы выберите **yes**.

55. Перетащите на созданную диаграмму из браузера классы **ВыборЗаказа** и **ДеталиЗаказа**.

56. В навигаторе выберите диаграмму **Главная (класс)**, выберите пакет **сущности**, **M2**, на запрос о создании диаграммы выберите **yes**.

57. Перетащите на нее из навигатора класс **Заказ**.

58. В навигаторе выберите диаграмму **Главная (класс)**, выберите пакет **управление**, **M2**, на запрос о создании диаграммы выберите **yes**.

59. Перетащите на нее из навигатора классы **МенеджерЗаказов**, **МенеджерТранзакций**.

### 3.8. Добавление атрибутов

Добавим атрибуты и операции к классам диаграммы классов **Ввод нового заказа**.

60. Установите параметры: выберите **Редактировать | Установки | Нотации**, выберите флажком **Разрешить только строгую UML нотацию**, **Использовать угловые кавычки для стереотипов**, **Show type and parameters**.

61. На диаграмме классов **Ввод заказа (класс)** выберите класс **Заказ**, в окне **Детали** нажмите кнопку **Новый атрибут**, в поле **Имя** введите **orderNumber : int**

62. В класс **Заказ** введите атрибуты **customerName : char**

### 3.9. Описание операций с помощью диаграммы Классов

63. На диаграмме классов **Ввод заказа (класс)** выберите класс **Заказ**, в окне **Детали** нажмите кнопку **Новая операция**, в поле **Имя** введите **create() : boolean**

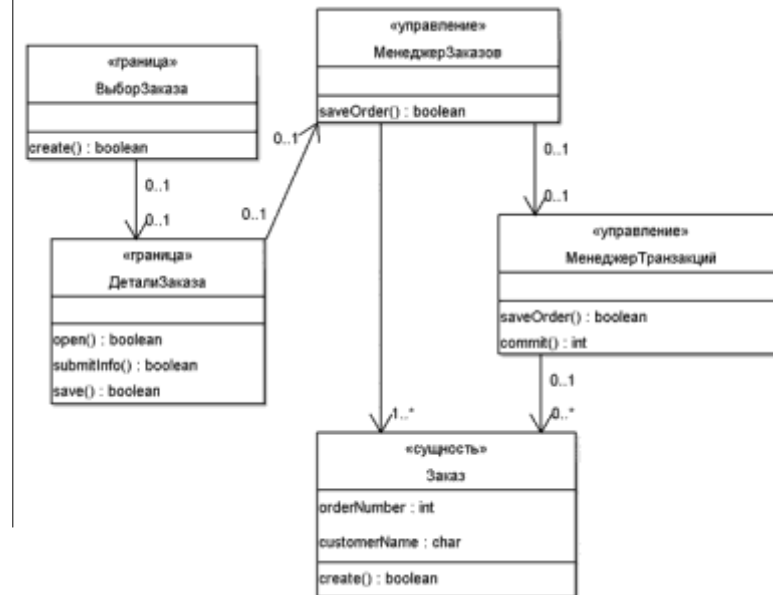
64. Для класса **Заказ** введите операции: **setInfo() : boolean** и **getInfo() : char**

65. Введите следующую сигнатуру операций класса **ДеталиЗаказа**: **open() : boolean**, **submitInfo() : boolean**, **save() : boolean**

66. Введите следующую сигнатуру операций класса **ВыборЗаказа**: **create() : boolean**

65. Введите следующую сигнатуру операций класса **МенеджерЗаказов**: **saveOrder() : boolean**

66. Введите следующую сигнатуру операций класса **МенеджерТранзакций**: **saveOrder() : boolean**, **commit() : int**



### 3.11. Добавление связей

Добавим связи к классам, принимающим участие в варианте использования **Ввести новый заказ**. 67. С помо-

щью кнопки **New UniAssociation** создайте на диаграмме **Ввод заказа (класс)** однонаправленные ассоциации по **Рис. 3**.

**68.** Добавьте на диаграмму значения множественности для ассоциаций по **Рис. 3**.

**69.** Как оформлять? — выберите в навигаторе **диаграмму вариантов использования 1, МП**, выберите **Copy Diagram to Clipboard**, запустите **MS Word**, выберите **Правка | Вставить**